



# «««ESERCITAZIONE»»»

## ESERCIZIO\_PRATICO\_U3\_W3\_

## L2

---



# TRACCIA

Lo scopo dell'esercizio di oggi è di acquisire esperienza con IDA, un tool fondamentale per l'analisi statica. A tal proposito, con riferimento al malware chiamato «Malware\_U3\_W3\_L2» presente all'interno della cartella «Esercizio\_Pratico\_U3\_W3\_L2» sul Desktop della macchina virtuale dedicata all'analisi dei malware, rispondere ai seguenti quesiti, utilizzando IDA Pro.

1. Individuare l'indirizzo della funzione DLLMain (così com'è, in esadecimale)
2. Dalla scheda «imports» individuare la funzione «gethostbyname». Qual è l'indirizzo dell'import? Cosa fa la funzione?
3. Quante sono le variabili locali della funzione alla locazione di memoria 0x10001656?
4. Quanti sono, invece, i parametri della funzione sopra?
5. Inserire altre considerazioni macro livello sul malware (comportamento)



## INDIVIDUARE L'INDIRIZZO DELLA FUNZIONE DLLMAIN (COSÌ COM'È, IN ESADECIMALE)

```
1000D02E 8B 44 24 08 48 0F 85 CE 00 00 00 8B 44 24 04 53 iD$.H.à+...iD
1000D03E A3 00 30 09 10 A1 44 90 01 10 56 83 C0 0D 57 50 ú.0..iDÉ..Uâ+
1000D04E E8 F9 7E 00 00 8B 1D 08 62 01 10 8B 35 C0 62 01 b"~...i..b...i5
1000D05E 10 33 FF 59 85 C0 74 23 A1 44 90 01 10 6A 07 83 .3 Vâ+t#iDÉ..
1000D06E C0 0D 68 58 39 09 10 50 FF D6 83 C4 0C 85 C0 75 +.hX9..P íâ-.
1000D07E 0A 57 57 57 68 74 10 00 10 EB 34 A1 44 90 01 10 .WWht...Ù4iD
1000D08E 83 C0 0D 50 E8 B5 7E 00 00 85 C0 59 74 2A A1 44 â+.PbÁ~..à+Yt
1000D09E 90 01 10 6A 06 83 C0 0D 68 4C 39 09 10 50 FF D6 É..j.â+.hL9..
1000D0AE 83 C4 0C 85 C0 75 11 57 57 57 68 65 13 00 10 57 â-.â+u.WWhe.
1000D0BE 57 FF D3 A3 04 30 09 10 A1 3C 90 01 10 8B 35 B8 W Éú.0..í<É..
1000D0CE 62 01 10 83 C0 0D 6A 10 50 68 D8 E5 08 10 FF D6 b..â+.j.PhÏÖ.
1000D0DE A1 38 90 01 10 6A 05 83 C0 0D 50 68 E8 E5 08 10 í8É..j.â+.Ph
1000D0EE FF D6 83 C4 18 57 57 57 68 56 16 00 10 57 57 FF íâ-.WWWhU...
1000D0FE D3 5F 5E A3 08 30 09 10 5B 6A 01 58 C2 0C 00 55 Ě^ú.0..[j.X-
1000D10E 8B EC 81 EC 00 03 00 00 53 56 57 33 F6 FF 75 08 iÿüý....SUV3+
1000D11E 56 68 2A 04 00 00 FF 15 28 62 01 10 BF 00 01 00 Uh*... .(b...
1000D12E 00 89 45 08 57 8D 85 00 FF FF FF 56 50 E8 12 7E .ëE.Wià. UP
1000D13E 00 00 FF 75 0C 8D 85 00 FF FF FF 50 E8 51 7E 00 .. u.ià. PH
1000D14E 00 8D 85 00 FF FF FF 50 E8 F1 7D 00 00 83 C4 18 .ià. Pb±}..
1000D15E 8D 5C 00 02 8D 85 00 FD FF FF 57 50 8D 85 00 FF i\..ià.z WPi
1000D16E FF FF 6A FF 50 56 56 FF 15 E4 61 01 10 6A 04 68 j PUU .õa..
1000D17E 00 10 00 00 53 56 FF 75 08 FF 15 E0 61 01 10 8B ....SU u. .Óa
1000D18E F8 56 8D 85 00 FD FF FF 53 50 57 FF 75 08 FF 15 °Uìà.z SPW u
1000D19E DC 61 01 10 68 60 4D 09 10 68 54 4D 09 10 FF 15 _a..h`M..hTM.
1000D1AE F8 60 01 10 50 FF 15 00 62 01 10 56 56 57 50 56 ò`..P ..b..UU

0000C42E 000000001000D02E: DllMain(x,x,x)
```

Function name	Segment
BlockInput	.text
CreateToolhelp32Snapshot	.text
DllEntryPoint	.text
DllMain(x,x,x)	.text
EnumProcessModules	.text
GetAdaptersInfo	.text
GetModuleFileNameExA	.text
HandlerProc	.text
ICClose	.text
ICCompress	.text
ICImageCompress	.text
ICOpen	.text

Per prima cosa per sapere quale sia l'indirizzo esadecimale della funzione dobbiamo andare a ricercare il nome della nostra funzione dall'elenco presente nella parte sinistra del programma IDA, come da immagine sopra.

Come potete vedere dall'immagine a sinistra, l'indirizzo esadecimale della funzione è **000000001000D02E**



DALLA SCHEDA «IMPORTS» INDIVIDUARE LA FUNZIONE «GETHOSTBYNAME». QUAL È L'INDIRIZZO DELL'IMPORT? COSA FA LA FUNZIONE?

Address	Ordinal	Name
00000000100163A8		waveInOpen
00000000100163AC		waveInClose
00000000100163B0		waveInUnprepareHeader
00000000100163B4		waveInPrepareHeader
00000000100163B8		waveInAddBuffer
00000000100163BC		waveInStart
00000000100163C4	18	select
00000000100163C8	11	inet_addr
00000000100163CC	52	gethostbyname
00000000100163D0	12	inet_ntoa

In questo caso per scoprire l'indirizzo di memoria dell'import chiamato <<*gethostbyname*>> dobbiamo recarci nella sezione **IMPORTS** e cercare la funzione desiderata come mostrato in figura, nel nostro caso l'indirizzo è **00000000100163CC**



# QUANTE SONO LE VARIABILI LOCALI DELLA FUNZIONE ALLA LOCAZIONE DI MEMORIA 0X10001656? QUANTI SONO, INVECE, I PARAMETRI DELLA FUNZIONE SOPRA?

```
var_675= byte ptr -675h
var_674= dword ptr -674h
hLibModule= dword ptr -670h
timeout= timeval ptr -66Ch
name= sockaddr ptr -664h
var_654= word ptr -654h
Dst= dword ptr -650h
Parameter= byte ptr -644h
var_640= byte ptr -640h
CommandLine= byte ptr -63Fh
Source= byte ptr -63Dh
Data= byte ptr -638h
var_637= byte ptr -637h
var_544= dword ptr -544h
var_50C= dword ptr -50Ch
var_500= dword ptr -500h
Buf2= byte ptr -4FCh
readfds= fd_set ptr -4BCh
phkResult= byte ptr -3B8h
var_3B0= dword ptr -3B0h
var_1A4= dword ptr -1A4h
var_194= dword ptr -194h
WSAData= WSAData ptr -190h
arg_0= dword ptr 4
```

Per rispondere alle domande poste nella traccia, dobbiamo andare ad aprire la funzione **SUB\_10001656**, quindi ci posizioniamo sulla scheda IDA View-A e ritorniamo sulla lista di funzioni che abbiamo visto prima e cerchiamo la funzione **SUB\_10001656**, quindi doppio click su di essa.

Il risultato del doppioclick lo trovate a sinistra, ovvero le informazioni riguardanti la nostra funzione:

Nel riquadro rosso abbiamo le variabili dato che come sappiamo in assembly i valori negativi sono le variabili locali mentre i valori positivi presenti nel riquadro arancione sono parametri



# CONSIDERAZIONI COMPORTAMENTO MALWARE

Andando a verificare gli import presenti nelle funzioni sembra che il malware vada ad agire sulle chiavi di registro, sui processi del sistema, sulla memoria e sul network del sistema compromettendo le corrette funzionalità di rete



# THANK YOU

